A Level Computer Science

Exam Style Questions

Unit 1.4.2

Data Structures

Stacks

Name		Date	

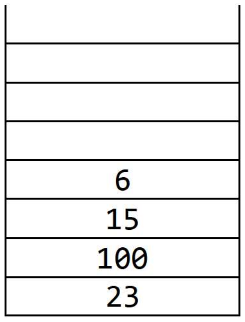
Score	Percentage	Grade
/ 27		

Question 1

A computer program stores data input on a stack named `dataItems`. The stack has two subprograms to add and remove data items from the stack. The stack is implemented as a 1D array, `dataArray`.

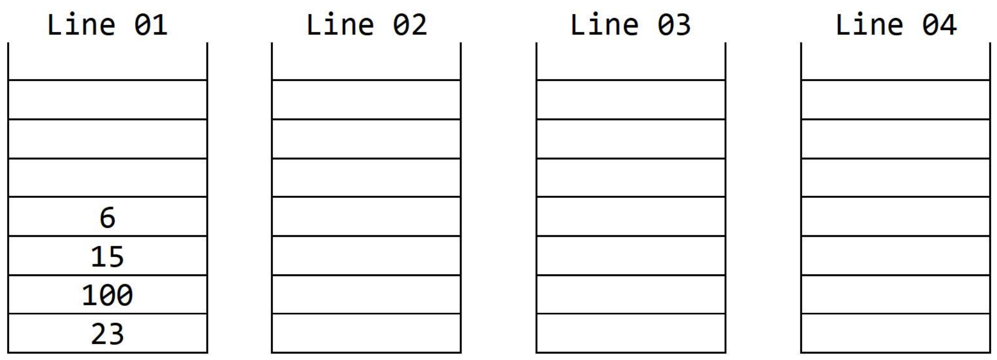
Sub-Program	Description
`push()`	The parameter is added to the top of the stack
`pop()`	The parameter is removed to the top of the stack

The current contents of `dataItems` are shown:



Show the contents of the stack `dataItems` after each of the following lines of code are run

```
01    push(13)
02    pop()
03    push(10)
04    push(20)
```



[4]

The main program asks a user to push or pop an item from the stack. If the user chooses 'push', the data item is added to the stack. If the user chooses "pop", the next item is removed from the stack, multiplied by 3 and output.

The main program is shown:

```
01    userAnswer = input("Would you like to push or pop an item?")
02    if userAnswer == "push" then
03        push(input("Enter data item"))
04    else
05        print(pop() * 3)
06    endif
```

Before the sub-programs, `push()` and `pop()`, can add or remove items from the stack, a selection statement is used to decide if each action is possible. Describe the decision that needs to be made in each sub-program and how this impacts the next process.

a) PUSH()

[2]

b) POP()

[2]

c) The algorithm does not work when the user enters "PUSH" or "Push". The algorithm needs to be changed in order to accept these inputs.

Identify the line number to be changed and state the change that should be made.

Line Number

[1]

Change

[1]

d) The stack is implemented as a 1D array, `dataArray`.
Describe how a 1D array can be set up and used to push and pop items as a stack.

[3]

Question 2

A user enters whole numbers into a computer program. Each number entered is placed onto a stack. The stack is created using an array with a maximum of 20 elements.

Part of the array, `numStack`, is shown when one number has been input.

index	stackItem
9	
8	
7	
6	
5	
4	
3	
2	
1	
0	20

top	1

The pointer, top, points to the next free space in the stack.

a) Complete the diagram below to show the state of `numStack` after the user inputs the following numbers in the order given:

22 13 2 59 1000

index	stackItem
9	
8	
7	
6	
5	
4	
3	
2	
1	
0	20

top	

[2]

b) A function, `addItem`, takes a number as a parameter and adds the number to the stack. The function returns `true` if this was successful, and `false` if the stack is already full.

i. Give one reason why a function is used instead of a procedure in this scenario.

[1]

ii. The function `addItem` is written but is incomplete.
Complete the function, `addItem`.

```
function addItem (number)

    if top == ____________ then

        return false
    else
        numStack [ ________ ] = ________

        top = ____________ + 1

        ______________________________

    endif

endfunction
```

[5]

iii. The procedure, `calculate`, takes each item in turn from the stack. It alternately adds then subtracts the numbers until there are none left.

For example, if `numStack` contains:

2
6
5
12

It would perform 2 + 6 – 5 + 12 and output 15.

```
01  procedure calculate()
02      total = 0
03      add = true
04      if top == 0 then
05          print("Stack empty")
06      else
07          total = numStack[top - 1]
08          top = top - 1
09          while top != 0
10              if add == true then
11                  total = total + numStack[top - 1]
12                  add = false
13              else
14                  total = total - numStack[top - 1]
15                  add = true
16              endif
17              top = top - 1
18          endwhile
19          print(total)
20      endif
21  endprocedure
```

Complete the trace table for the procedure `calculate`. The current array and pointer values when the procedure is called are on the first line of the trace table.

top	**numStack**						**total**	**add**	**output**
	0	**1**	**2**	**3**	**4**	**5**			
5	20	2	6	12	8				

[6]